



Progetto di Reti Logiche A: Non Maxima Suppression

Autori: Sabrina Cattaneo

Revisori: Marco D. Santambrogio

Data: 5/9/2007

Versione: 1.1

Stato: Final

Diffusione del documento

Documento interno al laboratorio di Microarchitetture, Dipartimento di Elettronica e Informazione, Politecnico di Milano.

Revisioni

Data	Versione	Stato	Commento
5/6/2006	1.0	Draft	Prima stesura del documento
5/9/2007	1.1	Final	

Si ringrazia per l'aiuto Chiara Sandionigi, Zullo Umberto e Santambrogio Marco.

Indice

<i>Revisioni</i>	3
<i>Indice</i>	4
<i>Indice delle figure</i>	5
<i>Indice delle tabelle</i>	5
<i>Introduzione</i>	6
Specifica del progetto.....	6
Edge detection.....	6
<i>Descrizione del lavoro</i>	8
Strumenti e ambiente di sviluppo.....	8
Struttura del componente.....	8
Comportamento del Core.....	11
Testbench.....	13
Risultati della sintesi.....	25
Conclusione e sviluppi futuri.....	25
<i>Bibliografia</i>	26

Indice delle figure

Figura 1 - Immagini rumorosa e filtrata a confronto.....	6
Figura 2 – Risultato dell’Edge Detection	7
Figura 3 - Pin out del core	8
Figura 4 – Dataflow dei componenti utilizzati per la simulazione	15
Figura 5 - Valore iniziale della matrice di pixel dell'immagine filtrata	16
Figura 6 – Valore iniziale della matrice di pixel dell'immagine risultate	16
Figura 7 – Pixel dell'immagine risultante al termine della simulazione (Edge Direction = 0)	20
Figura 8 – Pixel dell'immagine risultante al termine della simulazione (Edge Direction = 45)	20
Figura 9 – Pixel dell'immagine risultante al termine della simulazione (Edge Direction = 90)	21
Figura 10 - Pixel dell'immagine risultante al termine della simulazione (Edge Direction = 135)	21
Figura 11 - Temporizzazioni dei segnali del CORE (fase di acquisizione parametri)	22
Figura 12 – Temporizzazioni dei segnali del CORE (non-maxima-suppression di un pixel)	23
Figura 13 – Temporizzazioni dei segnali del CORE (fase di disattivazione)	24

Indice delle tabelle

Tabella 1 – Parametri per Int2	Error! Bookmark not defined.
Tabella 2 – Parametri per Int1	12
Tabella 3 – Valori di Ctrl per Int1	Error! Bookmark not defined.
Tabella 4 – Esempio di non-maxima-suppression su un pixel ..	Error! Bookmark not defined.
Tabella 1 - Parametri per Int2.....	12
Tabella 2 - Parametri per Int1.....	12
Tabella 3 - Valori di Ctrl per Int1	13
Tabella 4 - Esempio di non-maxima-suppression su un pixel.....	13
Tabella 5 - Metodo di calcolo delle posizioni dei pixel adiacenti	13
Tabella 6 - Non maxima suppression teorica per Edge Direction = 0.....	17
Tabella 7 - Non maxima suppression teorica per Edge Direction = 45.....	18
Tabella 8 - Non maxima suppression teorica per Edge Direction = 90.....	19
Tabella 9 - Non maxima suppression teorica per Edge Direction = 135.....	20

Introduzione

Specifica del progetto

Questo progetto ha lo scopo di realizzare un CORE definendone la struttura e funzionalità in linguaggio VHDL.

Il componente in questione dovrà eseguire la non-maxima-suppression, una delle mansioni in cui si suddivide l'algoritmo di Canny per l'esecuzione dell'edge detection.

Edge detection

L'edge detection è una procedura utilizzata dalle applicazioni di elaborazioni delle immagine per estrarre i contorni di un'immagine.

Esso consiste in :

- Smoothing, ovvero l'eliminazione del rumore (improvvisi sbalzi di intensità) per mezzo della convoluzione dell'immagine in un filtro gaussiano



Figura 1 - Immagini rumorosa e filtrata a confronto

- Calcolo del gradiente, attraverso il quale si ottiene, per ogni pixel, intensità e direzione del gradiente
- Non-maxima-suppression, che consiste nella soppressione dei pixel non massimi rispetto ai due adiacenti, ottenuti in base alla direzione del gradiente, sfruttando sia la funzione di smooting che quella per il calcolo del gradiente.

Hysteresis Threshold (il passo finale dell'algoritmo di Canny) ovvero la riduzione dell'immagine ad una matrice di pixel che possono essere "pixel di contorno" o "pixel non di contorno", è ottenuto attraverso l'analisi di ciascun pixel rispetto a due valori di soglia di intensità e la successiva verifica di appartenenza da parte dello stesso ad un tratto del contorno.

Il risultato finale dell'Edge Detection a partire dalla Figura 1 è visibile in Figura 2.



Figura 2 – Risultato dell'Edge Detection

Descrizione del lavoro

Strumenti e ambiente di sviluppo

Per la parte implementativa di questo progetto si è fatto uso di due prodotti software:

- ModelSim XE III 6.0a , ambiente di sviluppo e simulazione di componenti scritti in linguaggi VHDL. Esso comprende, oltre che un editor di testo, un compilatore VHDL e vari strumenti per la simulazione, tra cui un visualizzatore di forme d'onda e un'utilità che fornisce a video il pin-out dei segnali utilizzati nei processi desiderati con tanto di connessioni.
- Xilinx ISE 7.1i – Project Navigator , ambiente di verifica e sintesi dei componenti sviluppati in VHDL. Durante la fase di sintesi dei sorgenti VHDL di un componente ne viene verificata la realizzabilità in termini hardware: un componente simulabile infatti non è necessariamente sintetizzabile; di fatto quindi la sintesi introduce ulteriori limitazioni al linguaggio VHDL

Struttura del componente

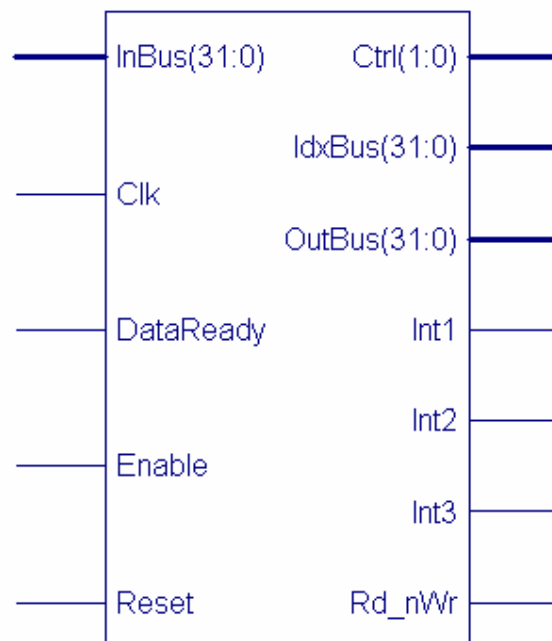


Figura 3- Pin-out del core

```

entity core is
  port
  (
    IdxBus      : out std_logic_vector(31 downto 0) bus;
    InBus       : in  std_logic_vector(31 downto 0) bus;
    OutBus      : out std_logic_vector(31 downto 0) bus;
    Ctrl        : out std_logic_vector( 1 downto 0) bus;

    Reset       : in  std_logic;
    Clk         : in  std_logic;

    -- INTERRUPT (Out)
    Int1        : out std_logic;
    Int2        : out std_logic;

    -- SEGNALI DI CONTROLLO
    DataReady   : in  std_logic;
    Rd_nWr     : out std_logic
  );
end entity core;
    
```

Il tipo di dato **std_logic** è un tipo di dato standard (definito nella libreria IEEE. std_logic_1164) utilizzato per rappresentare le connessioni tra componenti hardware. Esso deriva da un altro tipo di dato, **std_ulogic**, il quale rappresenta i più comuni casi verificabili nella manipolazione di un segnale logico:

```

TYPE std_ulogic IS ( 'U', -- Uninitialized
                    'X', -- Forcing Unknown
                    '0', -- Forcing 0
                    '1', -- Forcing 1
                    'Z', -- High Impedance
                    'W', -- Weak Unknown
                    'L', -- Weak 0
                    'H', -- Weak 1
                    '-'  -- Don't care
                    );
SUBTYPE std_logic IS resolved std_ulogic;
    
```

La differenza sostanziale tra **std_ulogic** e **std_logic** è che il secondo risolve i conflitti che possono verificarsi in caso di assegnamento multiplo, scegliendo per il segnale il valore assegnato superiore tra quelli in concorrenza. Questa risoluzione è effettuata dalla funzione **resolved**, definita nello stesso package di std_logic.

Di seguito vengono descritte le porte del core.

- Int1: Scrittura\Lettura di un pixel in un'immagine. Prima di porre Int1 a '1' si deve assegnare alle porte in uscita:

- InBus : valore del pixel da scrivere
 - Ctrl : ID dell'immagine su cui scrivere il pixel
 - IdxBus : indice del pixel da modificare (da 0 a "n° di pixel"-1)
 - Rd_nWr: '1' per leggere un pixel, '0' per scrivere un pixel
- Int2: Richiesta di un'informazione necessaria per eseguire la non_max_suppression. Prima di porre Int2 a '1' si deve assegnare alle porte in uscita :
- Ctrl : ID del dato richiesto (n° righe, n° colonne o edge direction)
 - Rd_nWr a '1'
- Ctrl è un bus utilizzato per comunicare all'interfaccia il parametro aggiuntivo per l'esecuzione di una richiesta del core (vedi Tabella 1 e 2)
- Int3: Impostato a '1' indica all'esterno che la non_max_suppression ha ultimato l'elaborazione sull'ultimo pixel dell'immagine.
- IdxBus indica all'interfaccia quale pixel leggere/scrivere da/a l'immagine specificata in Ctrl (IdxBus = 0 indica il primo pixel, IdxBus = 1 indica il secondo, e così via.)
- OutBus trasporta i dati in uscita. Nel caso del core contiene sempre il valore di un pixel da scrivere in un'immagine.
- Rd_nWr è un segnale di controllo. Se è 1 indica che è in corso il ciclo di lettura dal circuito di risposta.
- InBus trasporta i dati in ingresso. Nel caso del core può ricevere:
- Il valore di un pixel dall'immagine filtrata
 - Il numero di righe dell'immagine
 - Il numero di colonne dell'immagine
 - L'edge direction

Il core deve principalmente eseguire le seguenti operazioni:

1. Richiedere all'interfaccia i parametri necessari per l'esecuzione della non-maxima-suppression (numero di righe, numero di colonne, edge direction).

Questa operazione va eseguita interrompendo l'interfaccia tramite Int2. Di seguito vengono riportati i valori delle porte da impostare prima del settaggio dell'interruzione.

DATO DA PRELEVARE	Ctrl	Rd_nWr
Numero di colonne	0x00	1
Numero di righe	0x01	1
Edge direction	0x10	1

Tabella 1 - Parametri per Int2

2. Scorrere tutti i pixel dell'immagine risultante ed eseguire su ognuno di essi la non-maxima-suppression.

Per ogni pixel sono necessarie le seguenti elaborazioni

- Ottenimento del pixel adiacente sinistro dall'immagine filtrata
- Ottenimento del pixel adiacente destro dall'immagine filtrata
- Ottenimento del pixel attuale dall'immagine risultante
- Soppressione, a seguito del confronto tra i pixel ottenuti, del pixel attuale

Questa serie di operazioni si possono ridurre in due operazioni più generiche: la lettura di un pixel da un'immagine e la scrittura di un pixel da un'immagine.

Entrambe sono state fatte ricondurre ad un'unica interruzione (Int1), alla quale vanno abbinati i seguenti parametri:

AZIONE	Ctrl	IdxBus	OutBus	Rd_nWr
Scrittura di un pixel	Immagine di riferimento (vedi tabella 3)	Indice del pixel Da 0x00 a n° pixel -1	Valore da assegnare	0
Letture di un pixel			-	1

Tabella 2 - Parametri per Int1

Per entrambe le operazioni deve essere utilizzato uno tra questi valori di Ctrl:

IMMAGINE DI RIFERIMENTO	Ctrl
Immagine originale	0x00
Immagine filtrata	0x01

Immagine risultante	0x10
---------------------	------

Tabella 3 - Valori di Ctrl per Int1

La posizione dei pixel sinistro e destro adiacenti vengono determinate dalla posizione del pixel da esaminare e dall'edge direction.

In Tabella 4 è rappresentato l'esempio di un'immagine avente 6 righe e 6 colonne di pixel. Preso in esame il pixel (2 ; 2) dell'immagine risultante vengono evidenziati, per ogni valore di edge direction, i corrispondenti pixel adiacenti nell'immagine filtrata.

Immagine risultante						Immagine filtrata							
	0	1	2	3	4	5		0	1	2	3	4	5
0													
1									L	L	R		
2									L		R		
3									L	R	R		
4													
5													

L	Adiacente sinistro		Posizione pixel attuale
R	Adiacente destro		Edge direction = 0
			Edge direction = 45
			Edge direction = 90
			Edge direction = 135

Tabella 4 - Esempio di non-maxima-suppression su un pixel

Le posizioni dei pixel adiacenti si ottengono come segue:

EDGE DIRECTION	ADIACENTE SINISTRO	ADIACENTE DESTRO
0	Attuale - 1	Attuale + 1
45	Attuale + n° colonne - 1	Attuale - n° colonne + 1
90	Attuale - n° colonne	Attuale + n° colonne
135	Attuale - n° colonne - 1	Attuale + n° colonne + 1

Tabella 5 - Metodo di calcolo delle posizioni dei pixel adiacenti

Se il valore del pixel attuale è minore rispetto a quella di entrambi gli adiacenti il core richiederà all'interfaccia di azzerare il pixel attuale.

Testbench

Componenti utilizzati per la simulazione

Per verificare il buon funzionamento del core è stata predisposta un'entità di testbench, ovvero un componente fittizio senza uscite né ingressi, avente al suo interno dei componenti connessi tra loro, tra cui vi è il core.

Nel caso specifico sono stati connessi al core due componenti:

- Un generatore di clock e reset (clock_driver)
 - Sincronizzare le operazioni dei componenti a cui viene connesso.
 - Entro il secondo ciclo, inoltre, attiva e disattiva il segnale di Reset attivare gli stessi componenti.
- Un componente di risposta (core_responser) adibito all'esecuzione delle operazioni richieste dal core tramite Int1 , Int2 e Int3.
 - Come il core esso viene attivato a seguito della ricezione del segnale di Reset, mentre per il resto dell'elaborazione è posto in attesa di un'interruzione da parte del core.
 - Contiene inoltre i dati su cui il core esegue la non-maxima-suppression.

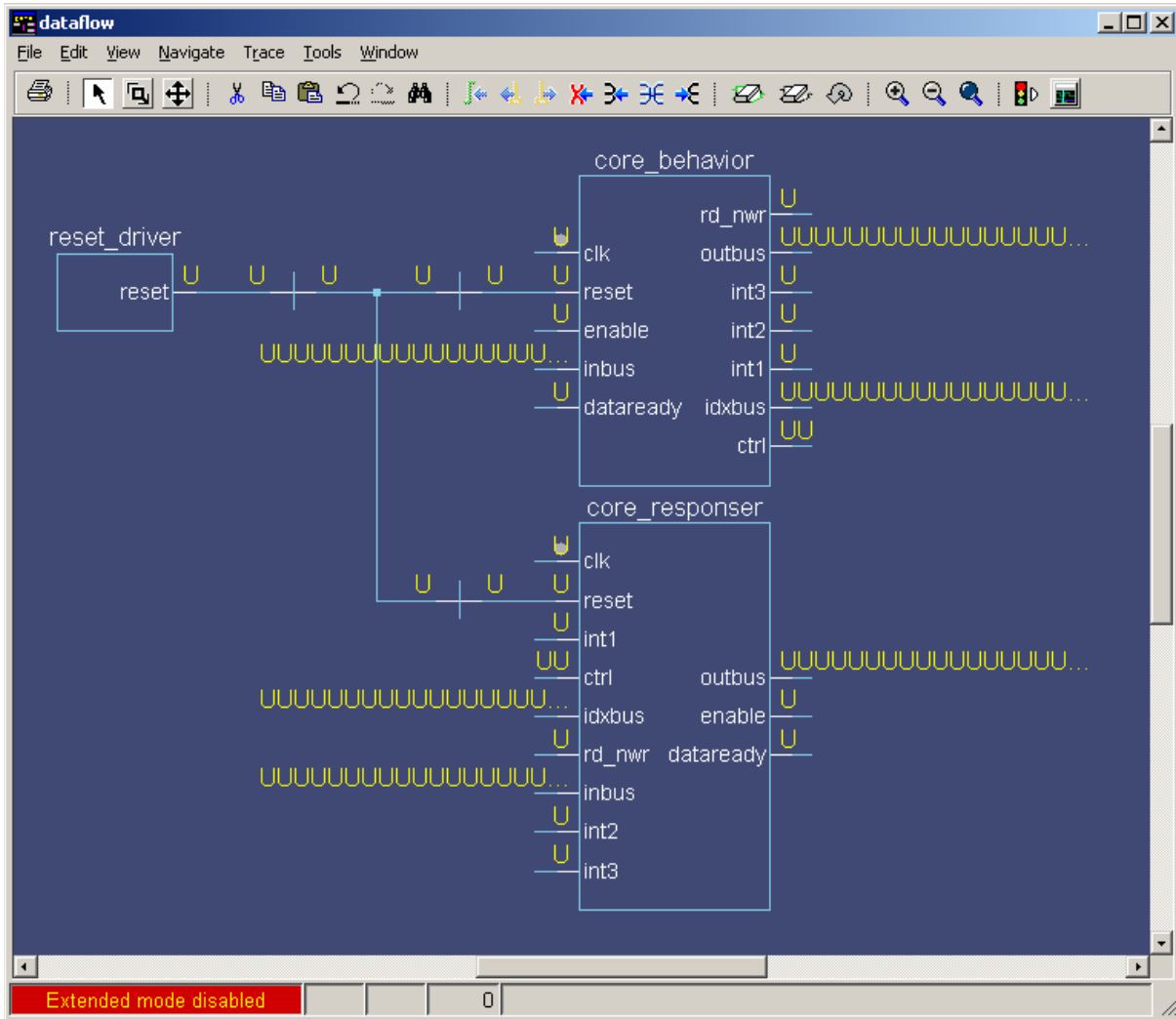


Figura 4 - Dataflow dei componenti utilizzati per la simulazione

Dati della simulazione

Il core lavora su due sole immagini: quella filtrata e quella risultante.

Per questa simulazione l'immagine filtrata (e quindi anche quella risultante) è costituita una matrice di 10 x 4 pixel.

Seguono i contenuti delle due immagini al momento dell'inizio della simulazione

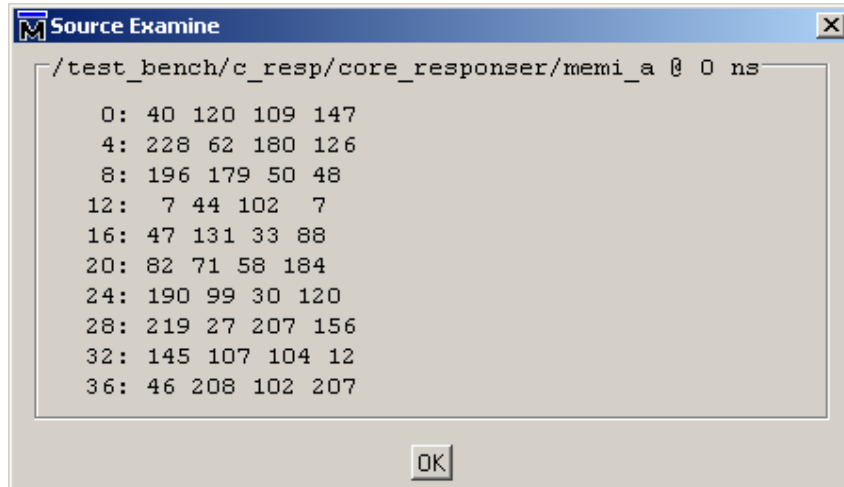


Figura 5 - Valore iniziale della matrice di pixel dell'immagine filtrata

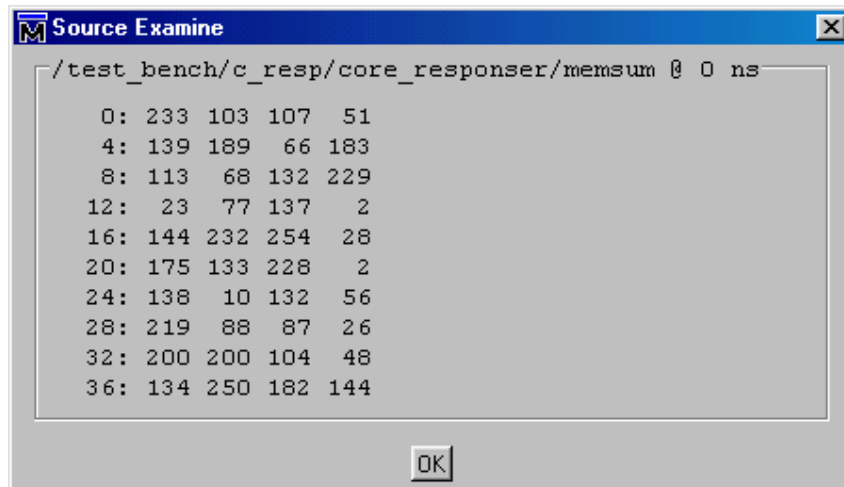


Figura 6 - Valore iniziale della matrice di pixel dell'immagine risultate

Dati teorici

Di seguono vengono riportati, per ogni valore di edge direction, i pixel contenuti nell'immagine risultante calcolati secondo le regole descritte nella Tabella 5.

I valori attuali superiori ad entrambi gli adiacenti sono evidenziati in rosso.

Coordinate pixel attuale (X;Y)	Valore pixel attuale	Valore pixel adiacente sinistro	Valore pixel adiacente destro
0;0	233		120
0;1	103	40	109
0;2	107	120	147
0;3	51	109	228
1;0	139	147	62
1;1	189	228	108
1;2	66	62	126
1;3	183	180	196
2;0	113	126	179
2;1	68	196	50
2;2	132	179	48
2;3	229	50	7
3;0	23	48	44
3;1	77	7	102
3;2	137	44	7
3;3	2	102	47
4;0	144	7	131
4;1	232	47	33
4;2	254	131	88
4;3	28	33	82
5;0	175	88	71
5;1	133	82	58
5;2	228	71	184
5;3	2	58	190
6;0	138	184	99
6;1	10	190	30
6;2	132	99	120
6;3	56	30	219
7;0	219	120	27
7;1	88	219	207
7;2	87	27	156
7;3	26	207	145
8;0	200	156	107
8;1	200	145	104
8;2	104	107	12
8;3	48	104	46
9;0	134	12	208
9;1	250	46	102
9;2	182	208	207
9;3	144	102	

Tabella 6 - Non maxima suppression teorica per Edge Direction = 0

Coordinate pixel attuale (X;Y)	Valore pixel attuale	Valore pixel adiacente sinistro	Valore pixel adiacente destro
0;0	233	147	
0;1	103	228	
0;2	107	62	
0;3	51	180	40
1;0	139	126	120
1;1	189	196	109
1;2	66	179	147
1;3	183	50	228
2;0	113	48	62
2;1	68	7	108
2;2	132	44	126
2;3	229	102	196
3;0	23	7	179
3;1	77	47	50
3;2	137	131	48
3;3	2	33	7
4;0	144	88	44
4;1	232	82	102
4;2	254	71	7
4;3	28	58	47
5;0	175	184	131
5;1	133	190	33
5;2	228	99	88
5;3	2	30	82
6;0	138	120	71
6;1	10	219	58
6;2	132	27	184
6;3	56	207	190
7;0	219	156	99
7;1	88	145	30
7;2	87	107	120
7;3	26	104	219
8;0	200	12	27
8;1	200	46	207
8;2	104	208	156
8;3	48	102	145
9;0	134	207	107
9;1	250		104
9;2	182		12
9;3	144		46

Tabella 7 - Non maxima suppression teorica per Edge Direction = 45

Coordinate pixel attuale (X;Y)	Valore pixel attuale	Valore pixel adiacente sinistro	Valore pixel adiacente destro
0;0	233		228
0;1	103		62
0;2	107		108
0;3	51		126
1;0	139	40	196
1;1	189	120	179
1;2	66	109	50
1;3	183	147	48
2;0	113	228	7
2;1	68	62	44
2;2	132	180	102
2;3	229	126	7
3;0	23	196	47
3;1	77	179	131
3;2	137	50	33
3;3	2	48	88
4;0	144	7	82
4;1	232	44	71
4;2	254	102	58
4;3	28	7	184
5;0	175	47	190
5;1	133	131	99
5;2	228	33	30
5;3	2	88	120
6;0	138	82	219
6;1	10	71	27
6;2	132	58	207
6;3	56	184	156
7;0	219	190	145
7;1	88	99	107
7;2	87	30	104
7;3	26	120	12
8;0	200	219	46
8;1	200	27	208
8;2	104	207	102
8;3	48	156	207
9;0	134	145	
9;1	250	107	
9;2	182	104	
9;3	144	12	

Tabella 8 - Non maxima suppression teorica per Edge Direction = 90

Coordinate pixel attuale (X;Y)	Valore pixel attuale	Valore pixel adiacente sinistro	Valore pixel adiacente destro
0;0	233		62
0;1	103		108
0;2	107		126
0;3	51		196
1;0	139		179
1;1	189	40	50
1;2	66	120	48
1;3	183	109	7
2;0	113	147	44
2;1	68	228	102
2;2	132	62	7
2;3	229	180	47
3;0	23	126	131
3;1	77	196	33
3;2	137	179	88
3;3	2	50	82
4;0	144	48	71
4;1	232	7	58
4;2	254	44	184
4;3	28	102	190
5;0	175	7	99
5;1	133	47	30
5;2	228	131	120
5;3	2	33	219
6;0	138	88	27
6;1	10	82	207
6;2	132	71	156
6;3	56	58	145
7;0	219	184	107
7;1	88	190	104
7;2	87	99	12
7;3	26	30	46
8;0	200	120	208
8;1	200	219	102
8;2	104	27	207
8;3	48	207	
9;0	134	156	
9;1	250	145	
9;2	182	107	
9;3	144	104	

Tabella 9 - Non maxima suppression teorica per Edge Direction = 135

Dati Sperimentali

Sperimentalmente si sono ottenuti i seguenti risultati

Time (ns)	Col 1	Col 2	Col 3	Col 4
0:	233	0	0	0
4:	0	0	0	0
8:	0	0	0	229
12:	0	0	137	0
16:	144	232	254	0
20:	175	133	228	0
24:	0	0	132	0
28:	219	0	0	0
32:	200	200	0	0
36:	0	250	0	144

Figura 7 - Pixel dell'immagine risultante al termine della simulazione (Edge Direction = 0)

Time (ns)	Col 1	Col 2	Col 3	Col 4
0:	233	0	107	0
4:	139	0	0	0
8:	113	0	132	229
12:	0	77	137	0
16:	144	232	254	0
20:	0	0	228	0
24:	138	0	0	0
28:	219	0	0	0
32:	200	0	0	0
36:	0	250	182	144

Figura 8 - Pixel dell'immagine risultante al termine della simulazione (Edge Direction = 45)

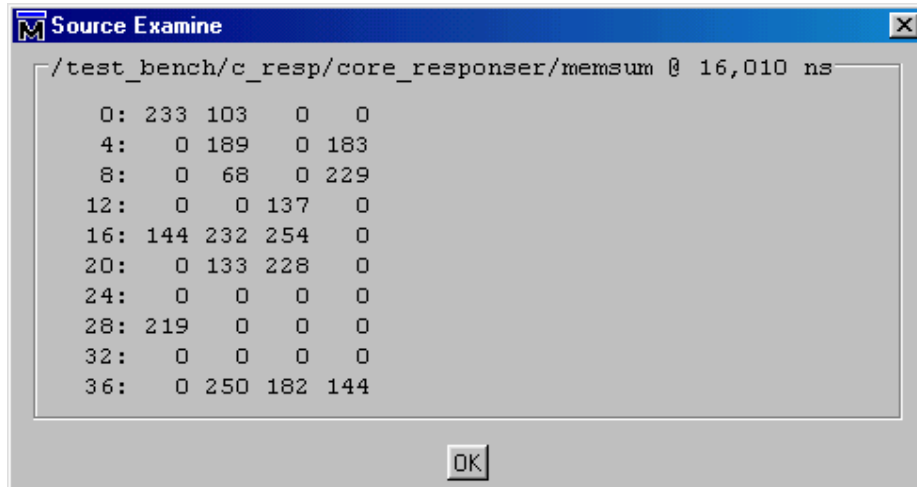


Figura 9 - Pixel dell'immagine risultante al termine della simulazione (Edge Direction = 90)

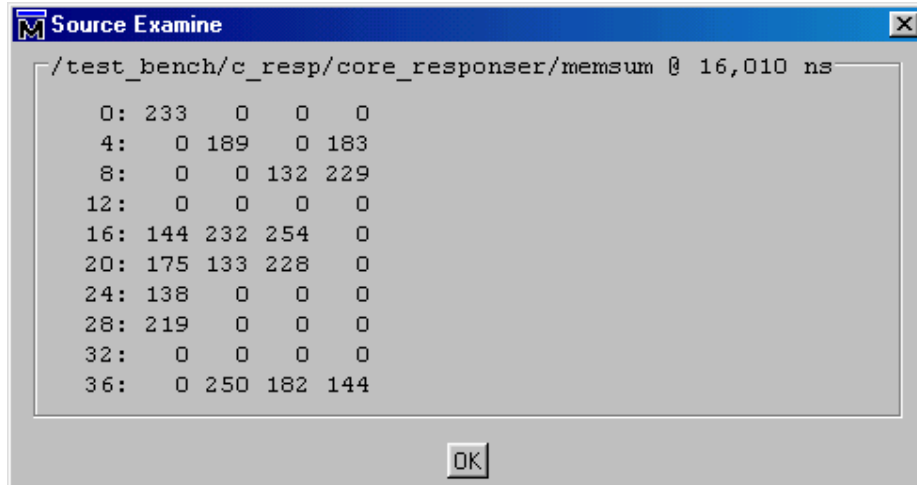


Figura 10 - Pixel dell'immagine risultante al termine della simulazione (Edge Direction = 135)

Temporizzazioni

In Figura 9 è riportato il timing dei segnali del CORE durante la fase di acquisizioni dei parametri per l'esecuzione della non-maxima-suppression:

1. Numero di colonne
2. Numero di righe
3. Edge direction

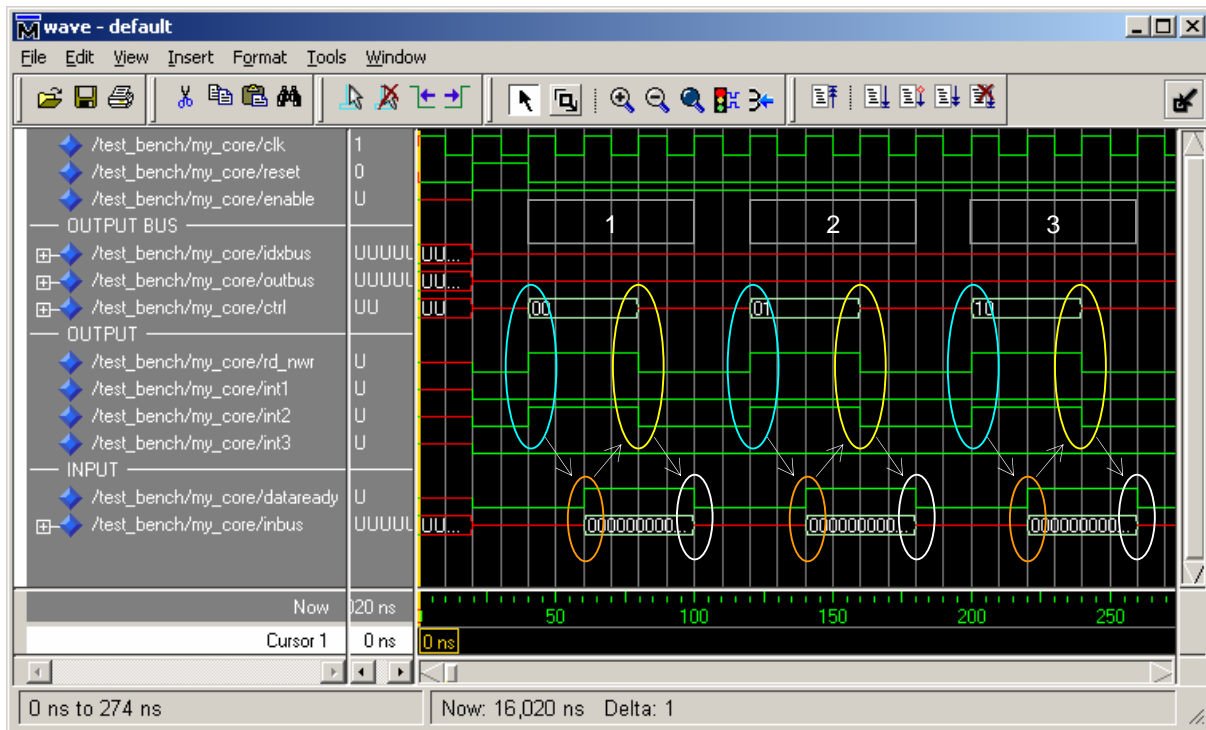


Figura 11 - Temporizzazioni dei segnali del CORE (fase di acquisizione parametri)

Questa parte dell'elaborazione ha luogo con CORE attivato (Enable='1') a seguito dell'azzeramento dei registri (Reset).

Le richieste dei dati vengono inoltrate tramite l'attivazione del segnale Int2 (vedi Tabella 1), e quindi hanno lo stesso comportamento nel tempo:

	Attivazione di Int2 e di Rd_nWr ; assegnamento a Ctrl dell'ID del dato da richiedere
	Ricezione del dato richiesto (su InBus) segnalata da DataReady
	Disattivazione di Int2 , Rd_nWr e Ctrl a seguito della lettura di InBus
	Disattivazione di DataReady e InBus da parte dell'interfaccia a seguito della disattivazione di Int2

In Figura 9 è riportato il timing dei segnali del CORE durante la fase di acquisizioni dei parametri per l'esecuzione della non-maxima-suppression:

1. Pixel sinistro da immagine filtrata
2. Pixel destro da immagine filtrata
3. Pixel attuale da immagine risultante
4. (Eventuale) azzeramento del pixel attuale dell'immagine risultante

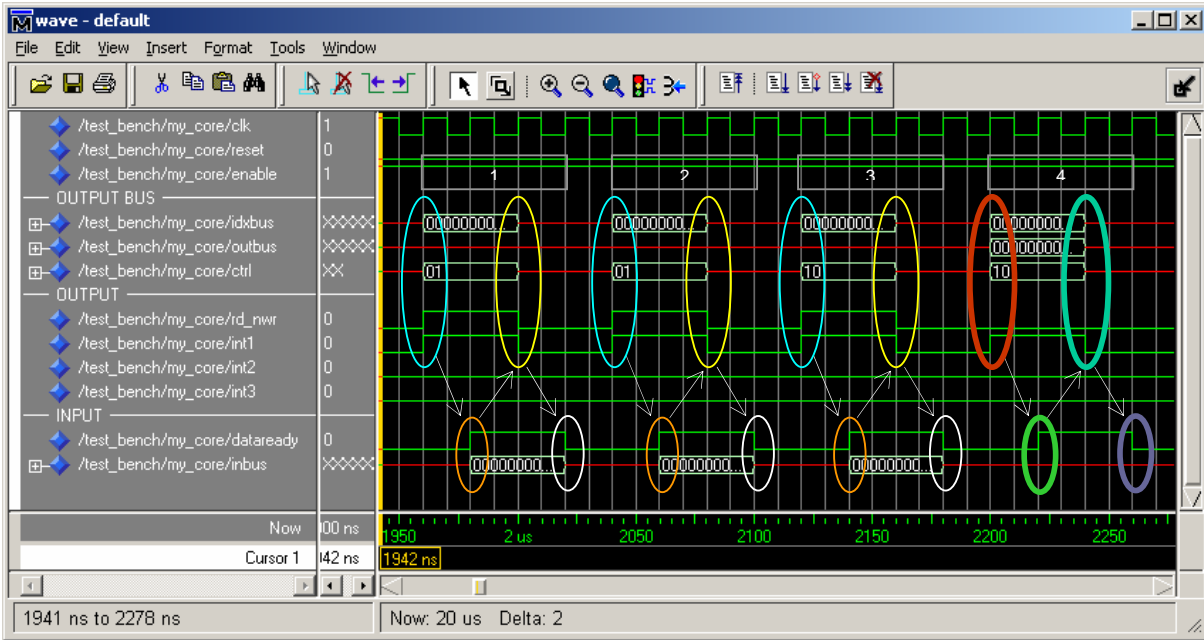


Figura 12 - Temporizzazioni dei segnali del CORE (non-maxima-suppression di un pixel)

Il CORE esegue la non-maxima-suppression sul primo pixel dell'immagine al termine dell'acquisizione dei parametri necessari per eseguire il ciclo.

Le quattro fasi della non-maxima-suppression sono risolte tutte tramite l'attivazione del segnale Int1 e dei relativi bus (vedi Tabella 2).

In particolare, come evidenziato in Figura 10, le fasi di acquisizione del valore di un pixel (1, 2 e 3) si svolgono in modo analogo:

- Attivazione di Int1 e di Rd_nWr e assegnamento dei parametri richiesti ai bus (vedi Tabella 2)
- Ricezione del valore del pixel richiesto (su InBus) segnalata da DataReady
- Disattivazione di Int1 , Rd_nWr , Ctrl e IdxBus a seguito della lettura di InBus
- Disattivazione di DataReady e InBus da parte dell'interfaccia a seguito della disattivazione di Int1

Di seguito è invece descritta la fase di scrittura di un pixel in un'immagine (4)

- Attivazione di Int1 ; Rd_nWr='0' ; assegnamento dei parametri richiesti ai bus (vedi Tabella 2)
- Ricezione della conferma della scrittura del pixel (segnalata da DataReady) da parte dell'interfaccia
- Disattivazione di Int1 , Ctrl e IdxBus a seguito della ricezione della conferma
- Disattivazione di DataReady da parte dell'interfaccia a seguito della disattivazione di Int1

In Figura 11 è rappresentato il timing dei segnali del CORE in corrispondenza della fase di disattivazione del core.

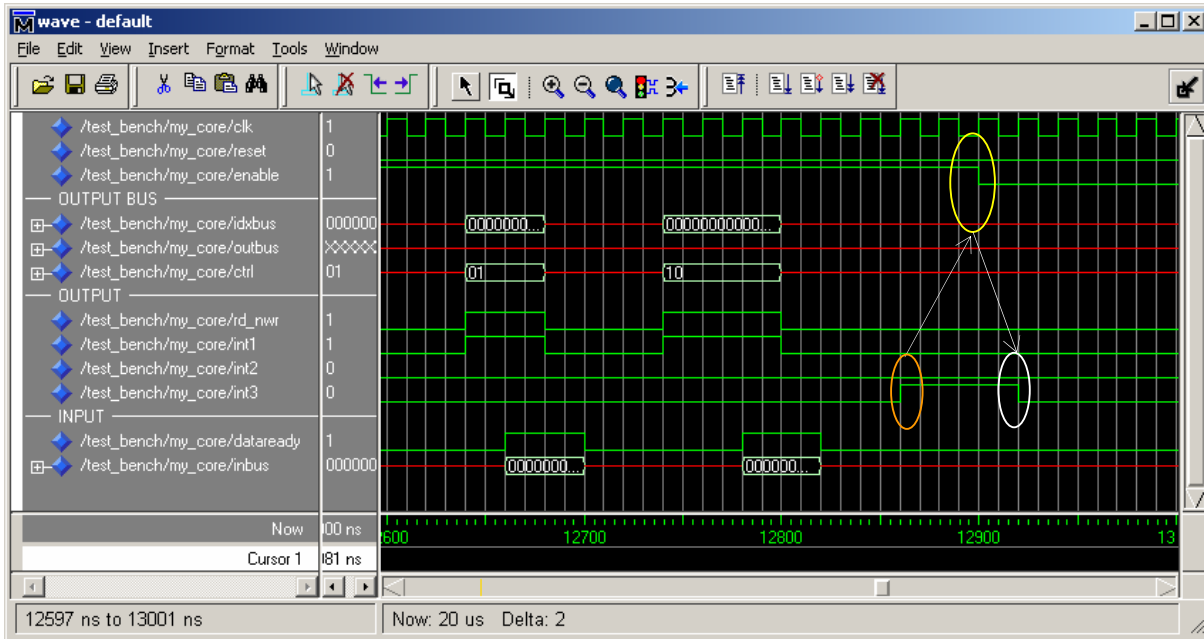


Figura 13 - Temporizzazioni dei segnali del CORE (fase di disattivazione)

Questa fase viene eseguita con l'attivazione di Int3, l'interruzione utilizzata per segnalare all'interfaccia che l'ultimo pixel dell'immagine è stata esaminata, e che quindi il core ha svolto la sua funzione.

A seguito della ricezione dell'interruzione, l'interfaccia disabilita il core attraverso il segnale Enable.

- Attivazione di Int3
- Disabilitazione del CORE da parte dell'interfaccia (attraverso il segnale Enable='0') a seguito della ricezione dell'interruzione
- Disattivazione di Int3 a seguito della disattivazione del CORE

Risultati della sintesi

Affinché l'IP Core possa essere importato e inserito all'interno dell'architettura, occorre che il VHDL che lo descrive sia sintetizzabile. Questo vuol dire che a partire dalla descrizione hardware del componente, deve essere possibile creare la rete che realizza la funzione descritta. Sfortunatamente il VHDL originariamente non era inteso come input di un tool di sintesi, e ciò comporta che non tutto il codice VHDL che si può scrivere può essere tradotto in hardware.

Dopo vari insuccessi soprattutto per la stesura di cicli oltre che simulabili anche sintetizzabili ha portato ad una modifica del flusso di lavoro che, se prima prevedeva il passo di sintesi solo dopo aver scritto e simulato il componente, a quel punto vedeva necessario un metodico alternarsi di simulazione e prove di sintesi, reso comunque facile dall'interoperabilità di Project Navigator e Modelsim.

Dopo vari tentativi si è giunti ad un codice che è anche sintetizzabile.

Conclusione e sviluppi futuri

Durante lo svolgimento del progetto è stata acquisita una notevole esperienza per quanto riguarda l'utilizzo degli strumenti ma soprattutto è servita ad una prima comprensione del funzionamento delle architetture integrate e delle problematiche legate al loro design. Possibili sviluppi futuri potrebbero consistere nello realizzare le altre funzioni dell'algoritmo di Canny e successivamente sintetizzarle.

Bibliografia

- [1] Alessandro Stranieri, Chiara Sandionigi "Come aggiungere una periferica in EDK", versione 6.3, 22 marzo 2005.
- [2] Xilinx "Embedded System Tools Guide", versione 6.2, 16 giugno 2004
- [3] Using and Creating Interrupt-Based Systems, xapp778 v1.0, 11 gennaio 2005.
- [4] Xilinx "Platform User Studio Guide", versione 6.3, 20 agosto 2004, cap. 5.
- [5] Xilinx "Edk PowerPC Tutorial", versione 6.2.
- [6] Xilinx "Synthesis and Verification Design Guide", all'interno di "Xilinx ISE 6 Software Manuals".
- [7] IEEE Standard for VHDL Register Transfer Level (RTL) Synthesis, IEEE Std 1076.6-1999.
- [8] Rishi R. Rakesh, Probal Chaudhuri e C. A. Murthy "Thresholding in Edge Detection: A Statistical Approach", IEEE TRANSACTIONS ON IMAGE PROCESSING